SETS

This page is intentionally blank

☺

INTRODUCTION

- Why *Mathematics*?
    - Natural language is not always *precise* and *unambiguous*
    - *Meaning* often depends on *context*

      Consider:

      He drives a red car
      He drives a hard bargain

      or:

      She sang like her sister
      She sang like a nightingale
      He sang like a canary

INTRODUCTION

- Mathematics
  - has a *proven track record* in science and engineering
  - is *precise*
  - is *concise* and self-contained
  - has *clarity* with little scope for misunderstanding
  - helps us concentrate on the *essentials*
  - is independent of natural language
  - may prove *correctness*

- We shall be concerned with the mathematics of *sets* and *logic* rather than numbers (though sets of *integer numbers* will be of interest)

DEFINITION & NOTATION

- A *set* is (informally) a:

  - well-defined,
  - unordered

  collection of ***similar*** items where each item is

  - *identifiable*, and
  - *distinct* from the other items

- A set may be defined by listing (or *enumerating*) its *members* or *elements* inside curly braces:

  {a, e, i, o, u} is the *set of vowels*, and

  {England, France, Ireland, Italy, Scotland, Wales}

  is *the set of countries which participate in Rugby Union's six-nations' championship*

SPECIAL SETS

- A set with just **one** member is a *singleton* set:

  {February} is *the set of months with less than 30 days*

- A set with no members is called the *null* set or *empty* set and is denoted either by {} or ∅

  e.g. the *set of all humans over twenty feet tall* is empty or null (i.e. = {})

- Certain sets of integers are denoted by generally accepted special symbols:

  $\mathbb{N}$ represents the set of *natural numbers* (≥ 0)

  $\mathbb{N}_1$ represents the non-zero *natural numbers* (≥ 1)

  and

  $\mathbb{Z}$ represents the set of positive and negative *integers* (i.e. whole numbers)

EQUIVALENCE & EQUALITY

- Two sets are *equal* or *equivalent* if, and only if, they have the same members

  e.g. {1, 2, 3, 5, 7, 11} and {3, 1, 2, 11, 5, 7} are equal/equivalent sets

- Set *definition* (i.e. *syntactic equivalence*) will be shown by = =

  e.g. *Vowels* = = {a, e, i, o, u}

  - Here *Vowels* is a shorthand 'name' for the set enumerated to the right of the = = sign

- The single = sign is often used to show equivalence between two sets but it may also be used to 'define' a set where the members of that set may change

  e.g. *CourseTeam* = {Smith, Jones, Patel}

SET MEMBERSHIP

- Set *membership* is denoted by $\in$ which is read as
  - *is a member of*, or
  - *is an element of*, or
  - *belongs to*

  e.g. u $\in$ *Vowels* (the set defined above)
  $0 \in \mathbb{N}$ and $-7 \in \mathbb{Z}$

- *Non-membership* is denoted by $\notin$ which is read as
  - *is not a member of*, or
  - *is not an element of*, or
  - *does not belong to*

  e.g. p $\notin$ *Vowels*
  $0 \notin \mathbb{N}_1$ and $3.14 \notin \mathbb{Z}$

OPERATIONS ON SETS - CARDINALITY

- The number of unique elements in a set is denoted by #

  e.g. $\#Vowels = 5$ and $\#\{\} = 0$

- For a set P, #P is often called the *size* or *cardinality* of the set P

- If #P is a finite number then P is said to be a *finite* set (otherwise it is an *infinite* set)

- Examples of infinite sets are: $\mathbb{Z}$, $\mathbb{N}$ and $\mathbb{N}_1$

- The arithmetic of infinite sets can seem 'weird':

  If A=={1, 2, 8} and B=={5, 7, 9, 17} then we can see #A = 3 and #B = 4 and so #B > #A

  What, though, of $\#\mathbb{N}$ and $\#\mathbb{N}_1$?

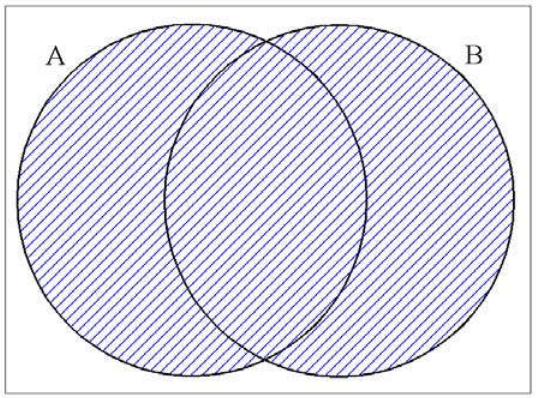
OPERATIONS ON SETS - UNION

- The *union* of two sets A and B is the set of all elements contained in both A and B with any element occurring in **both** A and B being listed **once only** in the union

- The *union* of A and B is written $A \cup B$.

- If A = = {p, q, u, v} and B = = {g, h, k, u, v, y}
  then $A \cup B$ = {g, h, k, p, q, u, v, y}

- A *Venn* diagram provides a graphic illustration with the union of sets A and B (i.e. $A \cup B$) depicted by the whole area shaded like 



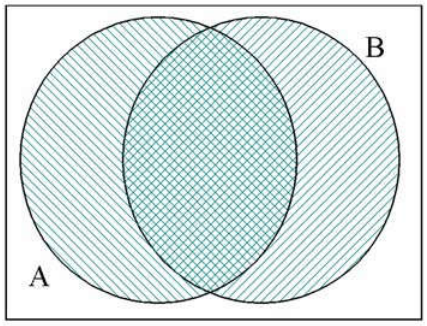
- The enclosing rectangle represents the *universal* set (i.e. **all** the elements in the domain in which we are interested)

OPERATIONS ON SETS - INTERSECTION

- The *intersection* of sets A and B is the set of those elements **common to both A and B** and is written $A \cap B$

- If A = = {p, q, u, v} and B = = {g, h, q, t, v, y}
  then $A \cap B = \{q, v\}$

- The corresponding *Venn* diagram might be as shown with the intersection of the two sets A and B (i.e. $A \cap B$) represented by the area shaded like 



- If A and B have no members in common they are said to be *disjoint* and we can then write:

$A \cap B = \{\}$ or $A \cap B = \varnothing$
or, less often, *disjoint* $\langle A,B \rangle$

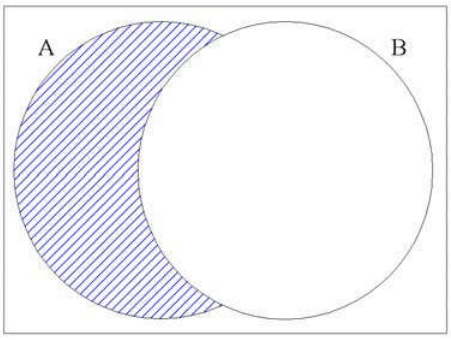
OPERATIONS ON SETS - DIFFERENCE

- The *difference* (or *relative complement*) of sets A and B is the set of all those elements which occur in A but not in B

- The *difference* of A and B is written A \ B

  e.g. if A = = {p, q, u, v} and
  B = = {g, h, k, u, v, y}
  then A \ B = {p, q}

- The corresponding *Venn* diagram might be as shown where A \ B is depicted by the area shaded similar to 



SUBSETS

- Suppose A = = {d, f, h, p, t} and B = = {h, t} then we notice that all members of set B are also members of set A. In such a case we say:
  - B is a *subset* of A, or
  - set B *is included in* set A

- If we know all members of set B are also in set A we can write $B \subseteq A$

- $B \subseteq A$ allows that the two sets *may*
  - be equivalent and
  - have exactly the same members

- An obvious corollary is that any set *must* be a subset of itself (i.e. for any set A, $A \subseteq A$)

- Note that (unlike $\in$) *set inclusion* ($\subseteq$) is transitive

SUBSETS

- If we know all of the members of B are also in A (i.e. $B \subseteq A$) but that A also has members which are not in B, then we should strictly write $B \subset A$

- For example:

$\{g, m\} \subseteq \{f, g, k, m, p\}$	is *true*
$\{a, p, t\} \subseteq \{p, t, a\}$	is *true*
$\varnothing \subseteq \{k, y\}$	is *true*
$\{a, p, t\} \subset \{p, t, a\}$	is *false*
$\{a, t\} \subset \{p, t, a\}$	is *true*

and, in particular:

$\mathbb{N}_1 \subset \mathbb{N}$	is *true*
$\mathbb{N} \subset \mathbb{Z}$	is *true*

POWERSETS

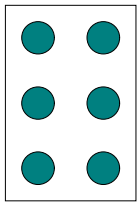
- The set of all possible subsets of a set A is called the *powerset* of A and is written $\mathbb{P}A$

- Since { } (i.e. $\varnothing$) is a valid subset of *any* set, the powerset of {a, b, c} is:

  $\{\{\ \}, \{a\}, \{b\}, \{c\}, \{a, b\}, \{b, c\}, \{c, a\}, \{a, b, c\}\}$

- An example of one of the uses of a powerset is the construction of *Braille* characters:

  - Each *Braille* character is based upon a 'cell' of 6 dots:



  - Each *Braille* "cell" can be regarded as a set of 6 possible dot-positions
    - The set of possible characters is the powerset of C (i.e. all possible subsets selected from C)
    - How *many* are there?

SET COMPREHENSION

- *Enumeration* is, generally, used to define sets only when there are not many members

- When *enumeration* is used
  - the set members can be analysed only by inspecting each and every member, and
  - even careful inspection does not always clarify the characteristic(s) shared by members of a set (e.g. consider {1, 3, 5})

- Sets are collections of objects which share similar characteristics and this fact provides a better mechanism (*abstraction*) for defining what a set contains

SET COMPREHENSION

- If *Big_Countries* is the set of countries with more than 100 million people and *C* represents the set of all countries, we could write:

  Big_Countries = = {c:*C*|c *has more than 100 million people*}

  - 'c:*C'* (the *signature*) means the values of c are drawn from the set *C*, and
  - the vertical bar '|' (called the *constraint* bar) is read as 'such that'

- The 'rule', 'condition' or 'constraint', appearing to the right of the *constraint bar* '|' is called a *predicate* and is either *false* or *true*

- The above way of specifying a set according to the characteristics shared by its members rather than by *enumeration* is called *set comprehension*

SET COMPREHENSION

- An alternative form of set comprehension specifies set elements by using a *pattern*

- If *Evens* is the set {0, 2, 4, 6, 8, ...} we can specify *Evens* as:

$$Evens == \{ x:\mathbb{N} \bullet 2x \}$$

which is read as:

'the set Evens is defined to comprise elements generated by the pattern or term "2 multiplied by x" where x is taken from the set of natural numbers'

- Similarly, if *Non_Zero_Evens* = = {2, 4, 6, ...}

we may write: $Non\_Zero\_Evens == \{x:\mathbb{N} \mid x>0 \bullet 2x\}$

where the constraining predicate ($x > 0$) acts as a filter to ensure non-zero values. Do we need it?

TYPED SETS

- Recall our (informal) definition of a set:

  A set is a well-defined, unordered collection of similar items where each item is clearly identifiable and distinct from the other items

- *Well-defined* means that given a 'value' we are able to decide whether it is a member of the set

- Sets are *homogeneous* in the sense that all members of a set are in some way similar

- All possible values that a set may have as members is said to define the *type* of the set

TYPED SETS

- If we attempted to model a *Library* system we would deal with sets of books and sets of people

- Suppose PERSON represents the set of all people that might ever be associated with our library, then

- At any given time, the set of Library staff

  - would be one particular *subset* of PERSON, and, hence

  - the set of Library staff, would be **one** of the sets defined by the powerset of PERSON

    Remember: *The powerset of a set A (i.e. ℙA) is the set of **all** possible subsets of A*

TYPED SETS

- Similarly, if BOOK represents the set of all books that might ever be associated with our library, then

- At any given time, the set of books actually on loan (or, similarly, the set of books available for loan)
  - would be one particular *subset* of BOOK, and, hence
  - at that time, the set of books on loan, say, would be one of the sets defined by the powerset of BOOK

TYPED SETS

- When using sets to specify systems we start by declaring *basic set types* (or *given sets*) which characterize the universal sets of objects we anticipate having to deal with

- These *basic types* are declared by writing them using upper-case letters in *square brackets*:

  [BOOK] and [PERSON] for a *Library* system
  [STUDENT] and [COURSE] for a *College Admin* system

  and, several types can be given in one declaration: [BOOK, PERSON]

- If each *member* of a set, which is based upon a given set, is of type T, then that set has type $\mathbb{P}$ T (remember, the *type* of the set is the set of *all possible* values the set may contain)

TYPED SETS

- In an academic *course-administration* system,
  - a **given set** could be [STUDENT]
  - *enrolled* could be the set of students who enrol on the BSc *Computing*, and
  - *graduated* could be the set of students who successfully complete the course
  - any *member* of each of the sets *enrolled* and *graduated* will be of type STUDENT
  - the type of **both** of the sets *enrolled* and *graduated* will be $\mathbb{P}$ STUDENT
  - we write: $enrolled, graduated : \mathbb{P}\ \text{STUDENT}$

- Equivalent statements to

  $$enrolled, graduated : \mathbb{P}\ \text{STUDENT}$$

  are:

  $enrolled \in \mathbb{P}\ \text{STUDENT}$; $graduated \in \mathbb{P}\ \text{STUDENT}$

  or:

  $enrolled \subseteq \text{STUDENT}$; $graduated \subseteq \text{STUDENT}$

FREE TYPES

- *Free types* or *enumerated types* can also be declared by enumerating the allowed identifiers for each of their elements:

  e.g. RESPONSE ::= yes | no
  STUDENT_GRADE ::= distinction | merit | pass | refer | fail

  where the vertical bar, "|", is read as "or"

- "RESPONSE ::= yes | no" is a shorthand for the following declarations and predicates:

[RESPONSE]	RESPONSE is a given set
yes : RESPONSE	*yes* is a value of the set
no : RESPONSE	*no* is a value of the set
yes ≠ no	*yes* and *no* are distinct
RESPONSE = {yes, no}	*yes* and *no* are the **only** values of the type

WELL-FORMED EXPRESSIONS

- When dealing with *typed* sets, the set operations considered previously (such as $\in$, #, $\cap$, etc) must only be applied to sets of compatible types

- If $Benelux == \{Belgium, Holland, Luxembourg\}$ and *Reference* is a set of books which may **not** be borrowed, then the members of the sets *Benelux* and *Reference* (and hence the sets themselves) are of *different* types

- It is, therefore, meaningless to write expressions such as

  $$Holland \notin Reference$$

  since *Holland* is **not of the same type** as the members of the set *Reference*

- Expressions involving **incompatible** types are said to be **not** *well-formed*

SUMMARY OF SET SYMBOLS

$\mathbb{Z}$	*Set of integers (positive or negative whole numbers)*
$\mathbb{N}$	*Set of natural numbers (≥ 0)*
$\mathbb{N}_1$	*Set of positive natural numbers (≥ 1)*
$t \in S$	*t is an element of set S*
$t \notin S$	*t is not an element of set S*
$S \subseteq T$	*Set S is contained in set T*
$S \not\subseteq T$	*Set S is not contained in set T*
$S \subset T$	*Set S is strictly contained in set T ($S \neq T$)*
$\varnothing$ or $\{\}$	*empty set*
$\{t_1, t_2, \ldots t_n\}$	*the set containing elements t_1, t_2, ... t_n*
$\mathbb{P}S$	*Powerset of set S : the set of all possible subsets of S*
$S \cup T$	*Union of sets S and T : the set of elements which are in S or in T or in both*
$S \cap T$	*Intersection of sets S and T : the set of elements which are both in S and in T*
$S \setminus T$	*Difference of sets S and T : the set of elements which are in S but not in T*
$\#S$	*Size or cardinality of set S : the number of elements contained in set S*
$\{D \mid P \bullet t\}$	*Set of elements t such that declarations D and P hold true*

EXERCISES

1. Translate the following *symbolic* statements into *English*:
   (a) $x \in S$
   (b) $x \notin S$
   (c) $X \subseteq Y$
   (d) $A \subset B$
   (e) $A \in B$
   (f) $B \notin C$

2. If NATO = = {*Belgium, Canada, Denmark, Iceland, Italy, Luxembourg, Holland, Norway, Portugal, United Kingdom, United States, Greece, Turkey, Spain, Germany*}
   EC = = {*Belgium, France, Germany, Italy, Luxembourg, Holland, Denmark, Greece, Ireland, United Kingdom, Spain, Portugal*}
   Scandinavia = = {*Denmark, Finland, Norway, Sweden, Iceland*}
   Benelux = = {*Belgium, Holland, Luxembourg*}
   Central America = = {*Costa Rica, Honduras, El Salvador, Guatemala, Nicaragua, Belize, Panama*},

   enumerate the following sets:

   (a) EC $\cup$ NATO
   (b) EC $\cap$ NATO
   (c) NATO \ EC
   (d) EC \ NATO
   (e) Scandinavia \ NATO
   (f) EC $\cup$ Benelux
   (g) EC \ Benelux
   (h) Benelux \ EC
   (i) NATO $\cap$ Scandinavia
   (j) Central America $\cap$ Benelux
   (k) EC $\cap$ (NATO $\cap$ Scandinavia)
   (l) (EC $\cap$ NATO) $\cap$ Scandinavia
   (m) EC $\cup$ (NATO $\cup$ Scandinavia)
   (n) (EC $\cup$ NATO) $\cup$ Scandinavia

3. Suppose A, B and C are three sets such that $A \subseteq B$ and $B \cap C$ is the empty set.
   (a) Draw a Venn diagram that illustrates this situation.
   (b) Draw another Venn diagram for the case when $A \subseteq B$ and $A \cap C$ is the empty set.

4. Use Venn diagrams to demonstrate
   (a) $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
   (b) $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$

5. In what circumstances will $A \setminus B = B \setminus A$?

6. The concepts of *union* and *intersection* of two sets can be extended to any number of sets. The symbols used are like those already encountered but are *larger* and and are written in front of the set of sets on which it operates. For example, the *union* of the sets {2, 5}, {2, 7, 8} and {3, 7, 8, 11} would be written:

$$\bigcup \{\{2, 5\}, \{2, 7, 8\}, \{3, 7, 8, 11\}\} = \{2, 3, 5, 7, 8, 11\}$$

   Similarly the intersection of the three sets is written:

$$\bigcap \{\{2, 5\}, \{2, 7, 8\}, \{3, 7, 8, 11\}\} = \{\} = \varnothing$$

   If A = {1, 2, 3, 4}, B = {2, 4, 6, 8, 19} and C = {2, 3, 5, 7} write down the sets:
   (a) $\bigcup \{A, B, C\}$
   (b) $\bigcap \{A, B, C\}$
   (c) $A \setminus (B \cap C)$

7. Given A = {a, b, c}, B = {b, c, d, e} and C = {a, b, c, d, e, f}, find:
   (a) #A
   (b) #C
   (c) $A \cup B$
   (d) $A \cap B$
   (e) $A \setminus B$
   (f) $A \cap (B \cup C)$
   (g) $A \cap (B \cap C)$

   State, giving reasons, whether the following statements are true for the above sets:

   (h) $b \in B$
   (i) $h \in B$
   (j) $A \in A$

8. Enumerate the following sets (e.g. $\{n : \mathbb{N} \mid n < 2\} = \{0, 1\}$):
   (a) $\{n : \mathbb{N} \mid n^2 < 17\}$
   (b) $\{n : \mathbb{Z} \mid n^2 < 17\}$
   (c) $\{n : \mathbb{N} \bullet n + 2\}$
   (d) $\{n : \mathbb{N} \mid n = 4 \bullet 2n\}$
   (e) $\{n : \mathbb{N} \mid n < 4 \bullet 2n\}$

9. Describe the following sets using set comprehension (there may be more than one answer!) (e.g. $\{0, 1, 8, 27\} = \{n:\mathbb{N} \mid n \leq 3 \bullet n^3\}$):

   (a) $\{0, 1, 2, 3, 4\}$
   (b) $\{0, 3, 6, 9, 12\}$
   (c) $\{1, 2, 4, 8, 16, 32\}$
   (d) The set of natural numbers greater than 15
   (e) The set of integers whose square is more than 40

10. Find the sets defined by
   (a) $\{n:\mathbb{N} \mid n > 7\} \cap \{n:\mathbb{N} \mid n < 10\}$
   (b) $\{n:\mathbb{N} \mid n > 7 \bullet n^2 + 4\} \cap \{\}$

11. Which of the following predicates are *true* and which are *false*?
   (a) $\{3, 4, 5\} \cup \{\} = \{\}$
   (b) $\{3, 4, 5\} \cap \{\} = \{\}$
   (c) $\{n:\mathbb{N} \mid n^2 < 20\} \cap \{1, 2, 3\} = \{2, 3\}$
   (d) $\{n:\mathbb{N} \mid n > 5 \text{ and } n < 10\} \cup \{n:\mathbb{N} \mid n \leq 5\} = \{n:\mathbb{N} \mid n < 10\}$

12. Set comprehension may be used to define *set union* as follows:
   Suppose the elements of the sets A and B are of type T, then the *union* of the sets A and B (written $A \cup B$) is defined by: $A \cup B = \{ x : T \mid (x \in A) \text{ OR } (x \in B)\}$
   or, if we use $\vee$ as a shorthand for OR: $A \cup B = \{ x : T \mid (x \in A) \vee (x \in B)\}$

   Using, where necessary, $\wedge$ for AND with $\vee$ for OR write set comprehensions for

   (a) set *intersection* ($\cap$)
   (b) set *difference* (\ )

13. Given the declarations: $x, y, z : \mathbb{Z}$; a : AUTHOR; b : BOOK;
   on_shelves : $\mathbb{P}$ BOOK; novelists : $\mathbb{P}$ AUTHOR
   say whether the following notations are *well-formed*
   (a) $x > y$
   (b) $x \in$ on_shelves
   (c) $a \in$ on_shelves
   (d) $a \in$ novelists
   (e) on_shelves $\subseteq$ novelists
   (f) on_shelves $\subseteq$ Book
   (g) {on_shelves, novelists}
   (h) {a, b}

14. Given the following definitions: $s = \{ 1, 4, 6, 7, 8, 9 \}$
$t = \{ x : \mathbb{N} \mid x^2 < 10 \bullet x^2 \}$

write down the results of the following expressions:

(a) $s \cup t$
(b) $s \cap t$
(c) $s \setminus t$

15. Suppose *int_sets* is a set containing sets of integers (i.e. $int\_sets : \mathbb{P}(\mathbb{P}\,\mathbb{N})$), with:

$$int\_sets = \{ \{1, 2, 3\}, \{ \}, \{3, 4, 5\}, \{3, 4\} \}$$

(a) What is produced by $\bigcup int\_sets$?
(b) What is produced by $\bigcap int\_sets$?

ANSWERS

1.

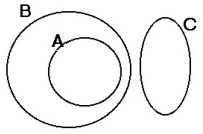
(a) *x is a member (or, an element) of the set S*
(b) *x is not a member of the set S*
(c) *The set X is a subset of the set Y,* or
The set X is included in the set Y, or
All members of the set X are also members of the set Y
(d) *The set A is a (proper) subset of the set B*, or
All members of the set A are also members of the set B, but there are members of B which do not occur in A
(e) *The set A is an element of (or a member of) the set B*
(f) *The set B is not an element of (or not a member of) the set C*

2.

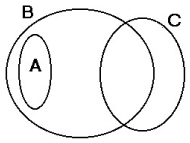
(a) {*Belgium, Canada, Denmark, Iceland, Italy, Luxembourg, Holland, Norway, Portugal, United Kingdom, United States, Greece, Turkey, Spain, Germany, France, Ireland*}
(b) {*Belgium, Denmark, Italy, Luxembourg, Holland, Portugal, United Kingdom, Greece, Spain, Germany*}
(c) {*Canada, Iceland, Norway, United States, Turkey*}
(d) {*France, Ireland*}
(e) {*Finland, Sweden*}
(f) Same as set EC
(g) {*France, Germany, Italy, Denmark, Greece, Ireland, United Kingdom, Spain, Portugal*}
(h) *Empty set*
(i) {*Denmark, Norway, Iceland*}
(j) *Empty set*
(k) {*Denmark*}
(l) *Same as (k)*
(m) {*Belgium, Canada, Denmark, Iceland, Italy, Luxembourg, Holland, Norway, Portugal, United Kingdom, United States, Greece, Turkey, Spain, Germany, France, Ireland*, *Finland, Sweden*}
(n) *Same as (m)*

3.

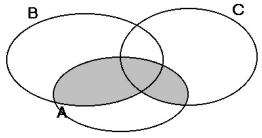
(a)

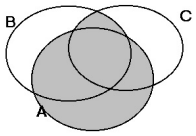


(b)



4.

(a) 

(b) 

5. *Only if* $A = B$

6.

(a) $\{1, 2, 3, 4, 5, 6, 7, 8, 19\}$
(b) $\{2\}$
(c) $\{1, 3, 4\}$

7.

(a) *3*
(b) *6*
(c) $\{a, b, c, d, e\}$
(d) $\{b, c\}$
(e) $\{a\}$
(f) $\{a, b, c\}$
(g) $\{b, c\}$
(h) *true, because b is explicitly listed as a member of set B*
(i) *false, because h is not explicitly listed as a member of set B*
(j) *false, because A is not explicitly listed as a member of set A*

8.

(a) $\{0, 1, 2, 3, 4\}$
(b) $\{-4, -3, -2, -1, 0, 1, 2, 3, 4\}$
(c) $\{2, 3, 4, 5, 6, 7, \ldots\}$
(d) $\{8\}$
(e) $\{0, 2, 4, 6\}$

9.

(a) $\{n:\mathbb{N} \mid n<5\}$
(b) $\{n:\mathbb{N} \mid n<5 \bullet 3n\}$
(c) $\{n:\mathbb{N} \mid n<6 \bullet 2^n\}$
(d) $\{n:\mathbb{N} \mid n>15\}$
(e) $\{n:\mathbb{Z} \mid n^2>40\}$

10.

(a) $\{8, 9\}$
(b) $\{\}$

11.

(a) *false*
(b) *true*
(c) *false*
(d) *true*

12.

(a) $A \cap B = \{ x : T \mid (x \in A) \wedge (x \in B)\}$
(b) $A \setminus B = \{ x : T \mid (x \in A) \wedge (x \notin B)\}$

13.

(a) *well-formed predicate*
(b) *not well-formed*
(c) *not well-formed*
(d) *well-formed predicate*
(e) *not well-formed*
(f) *well-formed predicate*
(g) *not well-formed*
(h) *not well-formed*

14. $t = \{ 0, 1, 4, 9 \}$; so

(a) $s \cup t = \{ 0, 1, 4, 6, 7, 8, 9 \}$
(b) $s \cap t = \{ 1, 4, 9 \}$
(c) $s \setminus t = \{ 6, 7, 8 \}$

15.

(a) { 1, 2, 3, 4, 5 }

(b) { }

SIMPLE SPECIFICATION WITH SETS

This page is intentionally blank

☺

INTRODUCTION

- With just the elementary mathematics of sets encountered so far we can describe a *simple* computerized system

- Of necessity, the system is trivial, but will enable us to give a taste of how mathematics could be used to specify a system:

  An aircraft has a fixed capacity and it is required to record the number of people aboard the aircraft at any time. The aircraft seats are not numbered and passengers enter the aircraft and choose seats on a first-come-first-served basis.

- Passengers belong to the set of all possible persons. Let this set be called *PERSON*.

- Hence, for this system, the *basic type* (or *given set*) is:

  [PERSON] the set of all possible uniquely identified persons

SYSTEM STATE

- If *capacity* denotes the seating capacity of the aircraft we have:

  $capacity : \mathbb{N}$ - the seating capacity of the aircraft

- At any time the *state* of the system is given by the number of passengers on the aircraft. We can describe this state by a set of persons, *onboard* (which will be one of the many possible subsets of *PERSON*)

- Hence we have: $onboard : \mathbb{P}\ \text{PERSON}$

- In addition we have the obvious constraint:

  $$\#onboard \leq capacity$$

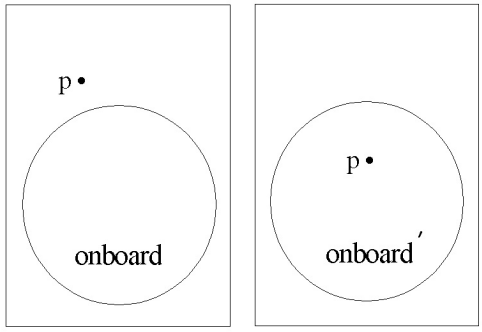
  - This constraint is an *invariant* for the system since, no matter what changes occur, it must still be true

OPERATIONS AFFECTING STATE

- Before passengers board the aircraft we will have the *initial state*:

  $onboard = \{\ \}$ (which satisfies the invariant since $\#\{\ \} = 0$)

- When a passenger, *p*, boards the aircraft the set *onboard* will change. The value of *onboard* after such a change is written as *onboard′* (read as "*onboard prime*")



Before ***After***

- Note: $onboard' = onboard \cup \{p\}$

  [also, by implication: $\#\ onboard' = \#\ onboard + 1$]

OPERATIONS AFFECTING STATE

- Clearly, for the invariant to be satisfied it is essential that before a passenger can board the aircraft, the following *precondition* must be met:

$$\# \mathit{onboard} < \mathit{capacity}$$

- If the person boarding is p then a further *precondition* must be:

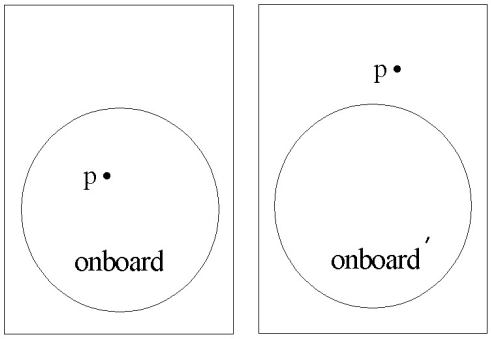
$$p \notin \mathit{onboard}$$

- Hence the boarding operation may be defined by

$$\begin{aligned} & p : \mathrm{PERSON} \\ & p \notin \mathit{onboard} \\ & \# \mathit{onboard} < \mathit{capacity} \\ & \mathit{onboard}' = \mathit{onboard} \cup \{p\} \end{aligned}$$

OPERATIONS AFFECTING STATE

- Passengers will also be able to disembark from the aircraft:



Before ***After***

- After disembarkation the following must be true:

$$onboard' = onboard \setminus \{p\}$$

with *precondition*: $p \in onboard$

- Summarizing: p **:** PERSON
  $p \in onboard$
  $onboard' = onboard \setminus \{p\}$

ENQUIRY OPERATIONS

- It would be useful to be able to determine how many people are on board the aircraft at any time. If this number is *numOnboard*, then

$$numOnboard : \mathbb{N}$$
$$numOnboard = \# onboard$$
$$onboard' = onboard$$

where the last statement clarifies that *onboard* does not change as a result of the query

- Another useful enquiry would tell whether a specific person is on board. Suppose the reply is a value of the *free type* RESPONSE, where:

$$RESPONSE ::= yes \mid no$$

Then, if p : PERSON; $reply$: RESPONSE

we have: $((p \in onboard \text{ AND } reply = yes)$
OR $(p \notin onboard \text{ AND } reply = no))$;

$$onboard' = onboard$$

FULL SYSTEM

- In summary the complete "state" description is:

[PERSON]	*The set of all possible uniquely identified persons*
$capacity : \mathbb{N}$	*The seating capacity of the aircraft*
$onboard : \mathbb{P}$ PERSON	*The set of persons on the aircraft (one of the many possible subsets of PERSON)*
$onboard \leq capacity$	*Constraint which is an invariant for system*
$onboard = \{\ \}$	*Initial state of the system*

Boarding is specified by:

p : PERSON	*p is a person*
$p \notin onboard$	*Precondition for embarkation*
$\#onboard < capacity$	*Precondition for embarkation*
$onboard' = onboard \cup \{p\}$	*True after p embarks on aircraft*

Disembarkation is specified by:

p : PERSON	*p is a person*
$p \in onboard$	*Precondition for disembarkation*
$onboard' = onboard \setminus \{p\}$	*True after p disembarks*

This page is intentionally blank

☺

EXERCISES

Scenario
A college provides a multi-user computer system for its students and staff. All staff and students must *register* with the college's IT Services unit before they are allowed access to the computer system. To use the system, each registered user must *log-in*. At any given time a registered user will either be *logged-in* or not *logged-in* (it is not possible for a user to be *logged-in* more than once concurrently).

Using the following declarations:

	[PERSON]	*the set of all uniquely identifiable persons*
users :	ℙ PERSON	*the set of all registered users*
logged_in :	ℙ PERSON	*the set of users who are currently logged-in*

express your solutions to the following symbolically (using sets) *and* in narrative form using **plain English.**

1. Discover any invariant properties of the system.

2. Define a suitable initial state for the system.

3. Define an operation to register a new user before their first *log-in*.

4. Define an operation to cancel a user's registration when the user is not *logged-in*.

5. Define operations for a registered user to *log-in* and to *log-out*.

(The example relating to 'passengers on an aircraft' should provide guidance)

LOGICAL PROPOSITIONS

This page is intentionally blank

☺

INTRODUCTION

- *Propositional logic* (aka *propositional calculus* or *Boolean algebra*) is concerned with statements (aka *propositions*) which may be *false* or *true* (but never both!)

  e.g. *Northampton is in Scotland*
  Tony Williams is a bellringer

- Simple propositions can be combined to form more complex propositions using *logical connectives*

- Typical logical connectives are:

  AND (symbolized by $\wedge$)
  OR (symbolized by $\vee$)
  NOT (symbolized by $\neg$)

- Logical connectives are also known as logical *operators* and the effect of such operators may be shown by a *truth table*

TRUTH TABLES - AND

- $\wedge$ or AND is often called the *conjunction* operator

- If P and Q are logical propositions then the truth table showing the effect of conjunction on P and Q is:

P	Q	P$\wedge$Q
false	*false*	false
false	*true*	false
true	*false*	false
true	*true*	true

- The table shows P$\wedge$Q is *true* if and only if P is *true* **and** Q is also *true*; otherwise it is *false*

  e.g. **(*July is the seventh month)*** $\wedge$ **(*July has 31 days*)**

  is *true* because **each** of the component propositions is, separately, *true*

TRUTH TABLES - OR

- $\vee$ or OR is often called the *disjunction* operator

- If P and Q are logical propositions then the truth table showing the effect of disjunction on P and Q is:

P	Q	P$\vee$Q
false	*false*	false
false	*true*	true
true	*false*	true
true	*true*	true

- This table shows that P$\vee$Q is *true* if either P is *true* **or** Q is *true* (**or both** P and Q are *true*); otherwise it is *false*

    - P$\vee$Q is **false** only if **both** P and Q are false

TRUTH TABLES - NOT

- ¬ or NOT is often called the *negation* operator

- If P is a logical proposition then the truth table showing the effect of negation on P is:

P	¬P
false	true
true	false

- ¬P always has the opposite "truth value" to P

  e.g. suppose we have the following propositions P, Q and R:

  P : *the book is on the library shelf*
  Q : *the student is allowed to borrow books*
  R : *the book is a reference copy*

  If $P \wedge Q \wedge (\neg R)$ is *true* (because each conjoined component is true) then it is also true that the student can borrow the book

OPERATOR PRIORITIES

- As in the example just given it is possible to use parentheses to clarify operator precedence (anything inside parentheses is evaluated *first*)

- Parentheses can always be used to clarify meaning but there is an agreed order of evaluation if a number of operators occur in a single proposition:

  $\neg$ has highest priority
  $\wedge$ has next highest priority
  $\vee$ has next highest priority

  e.g. for propositions P, Q and R

  $\neg P \vee Q \wedge R$ is evaluated as $(\neg P) \vee (Q \wedge R)$
  and $\neg P \wedge Q \vee R$ is evaluated as $((\neg P) \wedge Q) \vee R$

LOGICAL IMPLICATION

- If the truth of one proposition implies that another is also *true* we can use the implication operator $\Rightarrow$

- If P and Q are propositions then P $\Rightarrow$ Q is read as either
  - *if P then Q* or as
  - *P implies Q*

- the truth table for implication is:

P	Q	P $\Rightarrow$ Q
false	*false*	true
false	*true*	true
true	*false*	false
true	*true*	true

- P $\Rightarrow$ Q can only be *false* when P is *true* and Q is *false*

LOGICAL IMPLICATION

- Example of possible use:
  - quadrilateral is a square $\Rightarrow$ two adjacent sides are equal

    which may be read as:

    The quadrilateral may not be a square
    but, if it is,
    then two adjacent sides must be equal

  - the bulb lights $\Rightarrow$ the power supply is connected

    Which may be read as:

    The bulb may not light
    but, if it does,
    then the power supply must be connected

- It is possible to show with a truth table that the propositions $P \Rightarrow Q$ and $\neg P \vee Q$ are equivalent and, therefore, the symbol $\Rightarrow$ can, if preferred, always be eliminated from logical expressions

LOGICAL EQUIVALENCE

- The *equivalence* operator behaves as a *logical equality* operator and has the meaning of the English phrases *"exactly when"* or *"only when"* or *"if, and only if"*

- The symbol for *equivalence* is $\Leftrightarrow$ and the truth table for *equivalence* is:

P	Q	P $\Leftrightarrow$ Q
false	*false*	true
false	*true*	false
true	*false*	false
true	*true*	true

- P $\Leftrightarrow$ Q is equivalent to (P $\Rightarrow$ Q) $\wedge$ (Q $\Rightarrow$ P)

- Example of use:

valid_user_number $\wedge$ matching_password
$\Leftrightarrow$ login_successful

AIRCRAFT SPECIFICATION REVISITED

- We shall now illustrate a typical use of the logical operators by extending our specification of the *aircraft passenger system* considered previously

- In that system, we considered the operations for
  - *boarding* and
  - *disembarking*

- Both operations required *preconditions*; but we did not consider what was to happen if the preconditions were not satisfied

- This will now be remedied by introducing a free type FEEDBACK, where:

  FEEDBACK ::= OK | on board | full | not on board | two errors

  to provide a response from the system to clarify the outcome of each operation

BOARDING OPERATION

- As before, when a passenger, p, boards the aircraft the value of *onboard* will change, and the value of *onboard* after such a change is written as $onboard'$

- As previously, we have: p : PERSON
  and now: *reply* : FEEDBACK

  Using our previous definitions, the total 'Boarding' operation may be specified as:

$$
\begin{aligned}
&(p \notin onboard \wedge \#onboard < capacity \wedge \\
&\quad onboard' = onboard \cup \{p\} \wedge reply = \text{OK}) \\
&\vee \\
&(p \in onboard \wedge \#onboard = capacity \wedge \\
&\quad onboard' = onboard \wedge reply = \text{two errors}) \\
&\vee \\
&(p \in onboard \wedge \#onboard < capacity \wedge \\
&\quad onboard' = onboard \wedge reply = \text{on board}) \\
&\vee \\
&(p \notin onboard \wedge \#onboard = capacity \wedge \\
&\quad onboard' = onboard \wedge reply = \text{full})
\end{aligned}
$$

  which caters for the original preconditions not being satisfied!

DISEMBARKING OPERATION

- A similar specification is possible to describe a passenger disembarking:

$$p : \text{PERSON}$$
$$reply : \text{FEEDBACK}$$

$$(p \in onboard \land$$
$$onboard' = onboard \setminus \{p\} \land reply = \text{OK})$$
$$\lor$$
$$(p \notin onboard \land$$
$$onboard' = onboard \land reply = \text{not on board})$$

  which, again, caters for the preconditions not being satisfied!

- The above operation specifications are somewhat complicated but, later, we shall see that the Z specification language offers a more concise approach!

SUMMARY OF SYMBOLS

- Summary of logical symbols introduced:

false, true	*logical constants*
$\neg P$	*negation* : 'not P'
$P \wedge Q$	*conjunction* : 'P and Q'
$P \vee Q$	*disjunction* : 'P or Q'
$P \Rightarrow Q$	*implication* : 'P implies Q' or 'if P then Q'
$P \Leftrightarrow Q$	*equivalence* : 'P is logically equivalent to Q'

- The precedence order for the operators is (highest first):

$$\neg,\ \wedge,\ \vee,\ \Rightarrow,\ \Leftrightarrow$$

EXERCISES

In what follows, P, Q and R represent logical propositions

1. Write out the truth tables for the following expressions:
   (a) $P \vee \neg Q$
   (b) $\neg P \wedge Q$
   (c) $P \Rightarrow \neg Q$
   (d) $P \wedge (Q \vee \neg R)$
   (e) $P \Rightarrow (Q \Rightarrow R)$

2. Use a truth table to demonstrate that the proposition $(P \wedge Q) \vee R$ is logically equivalent to the proposition $(P \vee R) \wedge (Q \vee R)$ (i.e. $(P \wedge Q) \vee R \Leftrightarrow (P \vee R) \wedge (Q \vee R)$)

3. Show, using a truth table: $(P \Rightarrow Q) \Leftrightarrow (\neg P \vee Q)$

4. Another logical operator may be introduced which is usually called *exclusive-or* (written *XOR*) or *exclusive-disjunction* (written $\vee_e$). The disjunction operator ($\vee$) is an *inclusive-or* operator in the sense that $P \vee Q$ is true when P is true or Q is true or **both** P and Q are true, whereas the compound expression (P xor Q) is true only if P is true or Q is true but **not** if **both** P and Q are true. Use truth tables to demonstrate:

   (a) $P \text{ XOR } Q \Leftrightarrow (P \vee Q) \wedge \neg(P \wedge Q)$
   (b) $P \text{ XOR } Q \Leftrightarrow (P \wedge \neg Q) \vee (\neg P \wedge Q)$

5. The proposition $P \Leftrightarrow Q$ states that the propositions P and Q are equivalent. This may, or may not, be true. If, however, we know that P and Q will always be equivalent no matter what the circumstances, then it is, perhaps, more appropriate to use $\equiv$ (the identity symbol) rather than $\Leftrightarrow$. Use truth tables to demonstrate the following identities:

(a)	$P \wedge Q \equiv Q \wedge P$	Shows $\wedge$ is a *commutative* operator
(b)	$P \vee Q \equiv Q \vee P$	Shows $\vee$ is a *commutative* operator
(c)	$(P \wedge Q) \wedge R \equiv P \wedge (Q \wedge R)$	Shows $\wedge$ is an *associative* operator
(d)	$(P \vee Q) \vee R \equiv P \vee (Q \vee R)$	Shows $\vee$ is an *associative* operator
(e)	$P \vee (Q \wedge R) \equiv (P \vee Q) \wedge (P \vee R)$	Shows $\vee$ is a *distributive* operator
(f)	$P \wedge (Q \vee R) \equiv (P \wedge Q) \vee (P \wedge R)$	Shows $\wedge$ is a *distributive* operator
(g)	$\neg (P \wedge Q) \equiv \neg P \vee \neg Q$	Known as *De Morgan's Law*
(h)	$\neg (P \vee Q) \equiv \neg P \wedge \neg Q$	Another of *De Morgan's Laws*
(i)	$\neg \neg P \equiv P$	*Double negation* property

Propositions which are always true (such as those just considered) are called *tautologies*.

6. Let P, Q and R be the propositions:
   P: the membership is less than 20
   Q: all the members are men
   R: the maximum number of members is 50

   (a) Describe, in *English*, the meaning of the following propositions:

   (i) $P \wedge Q$
   (ii) $\neg R$

   (b) Using P, Q and R as defined, represent, *symbolically*, the proposition:

   There are at least 20 members and some of them are women

7. Suppose P represents the proposition "Claire is happy" and Q represents the proposition "Claire is rich". Write, in symbolic form:

   (a) Claire is poor but happy
   (b) Claire is neither rich nor happy
   (c) Claire is either rich or unhappy
   (d) Claire is either poor or is both rich and unhappy.

8. Show that the following are *tautologies* (see comment at end of question 5 above):
   (a) $P \Rightarrow (P \vee Q)$
   (b) $P \wedge Q \Rightarrow P$
   (c) $((\neg P) \wedge (P \vee Q)) \Rightarrow Q$
   (d) $((\neg P) \vee Q) \Leftrightarrow (P \Rightarrow Q)$

9. A college provides a multi-user computer system for its members. All members must *register* with the college's IT Services unit before they are allowed access to the computer system. To use the system, each registered user must *log_in*. At any given time a registered user will either be *logged-in* or not *logged-in* and it is not possible for a user to be *logged-in* more than once concurrently. In the following, express your solutions both symbolically (using sets and propositional logic), in the style outlined towards the end of the preceding notes, *and* in narrative form using **plain** English. Your solutions should cater for the necessary preconditions **not** being satisfied and be based upon a free type of the form:

   RESPONSE ::= OK | Already a user | Not a user | Logged in | Not logged in

   (a) Define an operation to register a new user.
   (b) Define an operation to cancel a user's registration.
   (c) Define an operation to *log-in*.
   (d) Define an operation to *log-out*.

10. The operators of propositional logic ($\neg \wedge \vee \Rightarrow \Leftrightarrow$) obey recognized precedence rules (refer to the chapter summary). If P, Q, R and S represent logical propositions:

(a) Without altering their essential meaning, simplify the following predicates by omitting as many parentheses as possible:

(i) $(\neg (((\neg R) \wedge P) \vee Q)) \Rightarrow R$
(ii) $(\neg P) \Rightarrow (((P \Rightarrow Q) \Rightarrow R) \wedge S)$

(b) Insert parentheses to emphasise how the following predicates are interpreted according to the conventional precedence rules:

(i) $P \Rightarrow Q \Leftrightarrow \neg Q \Rightarrow \neg P$
(ii) $P \vee Q \wedge \neg R \vee Q \wedge P$

(c) Use a truth-table to establish the validity of De Morgan's law:

$$\neg (P \vee Q) \Leftrightarrow \neg P \wedge \neg Q$$

(d) Any logical proposition which is always *true* is called a *tautology*; any which is always *false* is called a *contradiction*. Identify which of the following propositions is a tautology, a contradiction or neither:

(i) $\neg (P \vee Q)$
(ii) $false \wedge \neg (P \vee Q)$
(iii) $false \vee true$

(e) Consider the following information:

Oscar either cycles to work or uses his car. If it is not raining, Oscar cycles to work. If it is raining then Oscar uses his car unless the car does not start, in which case he has to cycle to work in the rain unless he can get a push-start from his neighbour.

If P, Q and R represent propositions as follows:

P : a push-start is available;
Q : the car starts;
R : it is raining

write down a logical expression involving P, Q and R which evaluates to *true* if Oscar cycles to work and *false* otherwise.

11. Suppose [PERSON] is a given type representing the set of all people there might ever be, past, present or future. Let the following sets be defined:

men	:	the set of all men
women	:	the set of all women

and suppose a company (which employs only men and women) comprises only the following departments:

marketing	:	the set of all people working in the marketing department of the company
personnel	:	the set of all people working in the personnel department of the company
production	:	is the set of all people working in the production department of the company

(a) Define a suitable **type** common to all five sets.

(b) Use set notation with appropriate logical connectives to represent each of the following statements symbolically:

(i) people at the company are either men or women but not both;
(ii) each employee of the company is in precisely **one** of the three departments, *marketing*, *personnel* or *production*;
(iii) the *personnel* department has a maximum of 10 employees;
(iv) all the employees in the *marketing* department are women;
(v) the company employs more men than women.

(c) If we assume, instead, that each employee of the company can be in more than one department, write symbolic expressions to represent the following:

(i) the number of women who work in all three departments;
(ii) the number of men who work in *marketing* and *personnel* but not *production*.

12.

(a) The disjunction operator ($\vee$) is an *inclusive-or* logical connective in the sense that $P \vee Q$ is true when P is true or Q is true or **both** P and Q are true. An *exclusive-or* logical connective, $\otimes$, say, may be defined so that $P \otimes Q$ is true only if P is true or Q is true but **not** if **both** P and Q are true.

(i) Construct an appropriate truth-table for $P \otimes Q$

(ii) Show $P \otimes Q \Leftrightarrow (P \vee Q) \wedge (\neg (P \wedge Q))$

(b) A special logical connective, represented by ◊, has the following truth table:

P	Q	P◊Q
true	*true*	*false*
true	*false*	*false*
false	*true*	*false*
false	*false*	*true*

Show: $P \wedge Q \Leftrightarrow (P \lozenge P) \lozenge (Q \lozenge Q)$

13. All recognized modules which students at a university can study are modelled by the set *modules*. Modules that are taken in the first year are modelled by the set *firstYear*; those that are taken in the second year by the set *secondYear;* and those in the third year by the set *thirdYear*.

Express each of the following statements using set notation (note that the statements may not be consistent with each other).

(a) The total number of recognized modules that are available to be studied will never exceed 50.
(b) None of the available modules can be taken in different years of a course (i.e. every module can be taken only in the first year or only in the second year or only in the third year - where "or" is exclusive).
(c) The module *computing_fundamentals* is taught in the first year.
(d) The *computer_architecture* module may be taught in either year two or year three but never in both years.
(e) All modules taken in years one, two or three are recognized by the university.

RELATIONS

This page is intentionally blank

☺

INTRODUCTION

- *Relations* enable us to investigate connections between members of different sets even if those members are of different types
    - we may have a *set of students* and a corresponding *set of assignment grades*, or
    - a *set of cars* and an associated *set of parking-places*

- More specifically, if we were developing a *Library* system, then, to keep track of *who* has borrowed *what*, we would need to investigate the association between values of type BOOK and values of type PERSON

- Essentially, we need to investigate what are called *ordered-pairs*

ORDERED-PAIRS

- If we have types PERSON and BOOK with:

  Zen and the Art of Motor-Cycle Maintenance : BOOK

  Thomas Tallis : PERSON

  we can create an *ordered-pair* whose first member is the 'book' and whose second member is the 'person'

- The notation for such an **ordered** pairing is:

  Zen and the Art of Motor-Cycle Maintenance ↦ *Thomas Tallis*

  or, alternatively

  (*Zen and the Art of Motor-Cycle Maintenance, Thomas Tallis*)

- The ↦ 'arrow' is called a *maplet* and emphasises the *asymmetric* nature of an ordered-pair

- In general, each element of an ordered-pair may be of a different type and hence the type of an ordered-pair cannot, in general, be the type of either component

CARTESIAN PRODUCTS

- Given any two sets, we can, in general, form a third set consisting of *all* the ordered-pairs of elements from those two given sets

- This method of deriving such sets of ordered-pairs is called the *Cartesian product*

- If A is the set {5, 7, 9} and B is the set {3, 5}, then the Cartesian product of the sets A and B is the set of ordered-pairs:

  $$\{(5, 3), (5, 5), (7, 3), (7, 5), (9, 3), (9, 5)\}$$

- The symbol for a Cartesian product is ×, so we can write:

  $$A \times B = \{(5, 3), (5, 5), (7, 3), (7, 5), (9, 3), (9, 5)\}$$

  - $A \times B$ is read as "A cross B"
  - N.B. $A \times B \neq B \times A$

RELATIONS

- Suppose we have:

  [COUNTRY] the set of all countries
  [LANGUAGE] the set of all languages

  then we can define a *speaks* relation between each country and a language spoken in that country, and this relation can be shown as a set of ordered-pairs:

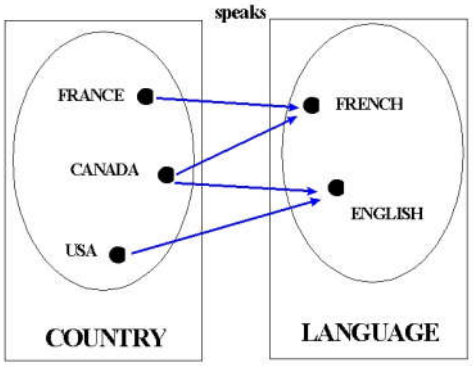
  e.g. {(France, French), (Canada, English), (Canada, French), (USA, English)}

- Note that the elements of the two sets are paired on a *many-to-many* basis; not just a *one-to-one* basis

- Such relations can be illustrated by diagrams called *directed graphs* (or *digraphs*)

SHOWING RELATIONS GRAPHICALLY

- A *digraph* to illustrate the *speaks* relation between the set of all countries and the set of all languages might be as shown alongside



- The set of ordered-pairs corresponding to the relation *speaks* could be declared as

  speaks : ℙ (COUNTRY × LANGUAGE)

- ℙ (COUNTRY × LANGUAGE)
  - defines all possible sets of ordered-pairs where the first element in each pair is a "country" and the second a "language"
  - represents the **type** of the *speaks* relation

- And we could similarly relate *language* to *country* by declaring a relation

  spoken : ℙ (LANGUAGE × COUNTRY)

  (but see later note on inverse relations)

RELATION SYMBOLS

- The notation just introduced is precise and perfectly acceptable but it is usual to denote a relation using a double-headed arrow: $\leftrightarrow$

- Using this notation the previously introduced relations would be written:

  $speaks : \text{COUNTRY} \leftrightarrow \text{LANGUAGE}$

  $spoken : \text{LANGUAGE} \leftrightarrow \text{COUNTRY}$

  - $\leftrightarrow$ is called the *relation symbol*

- For two sets X and Y, writing a type as $X \leftrightarrow Y$ means that, strictly, the type is $\mathbb{P}(X \times Y)$

- A statement such as

  $speaks : \text{COUNTRY} \leftrightarrow \text{LANGUAGE}$

  might be read as:

  "*speaks* relates *country* to *language*"

WAYS OF WRITING RELATIONS

- We have seen that a *relation* is a *set of ordered-pairs*

- We have already seen that ordered-pairs may be represented using either

  - the *maplet* arrow symbol, $\mapsto$, or
  - parenthesised values separated by a comma

- Hence, for the *speaks* relation, we can write the same information in different ways

  e.g. United Kingdom $\mapsto$ English $\in$ *speaks*
  = = (United Kingdom, English) $\in$ *speaks*

  and {(France, French), (Canada, English),
  (Canada, French), (USA, English)}
  = = {France $\mapsto$ French, Canada $\mapsto$ English,
  Canada $\mapsto$ French, USA $\mapsto$ English}

WAYS OF WRITING RELATIONS

- It is, also, acceptable to use English-like language to express a relation:
  - Austria *speaks* German
  - here the name of the relation is used as an *infix* operator between the two values of the ordered-pair
  - in this approach, the relation should be specified as _*speaks*_, with the underscores acting as "placeholders" for the operands

- In general, if R denotes a relation and values x and y are connected through the relation R, then:

$$x \mapsto y \in R == (x, y) \in R$$

but, if we define the relation using: _R_
then we can write: $x \ R \ y$

ELEMENTS OF RELATIONS

- To discover whether two values are related, it is sufficient to see if the pair is an element of the relation in question

  - If country *Austria* is linked to language *German* in the relation *speak*s where

    $$speaks : \mathbb{P}\ \text{COUNTRY} \times \text{LANGUAGE}$$

    then

    $Austria \mapsto German \in speaks$ will be **true**

  - and, if country *Austria* is linked to language *German* in the relation _*speak*s_ where

    $$\_speaks\_ : \mathbb{P}\ \text{COUNTRY} \times \text{LANGUAGE}$$

    then

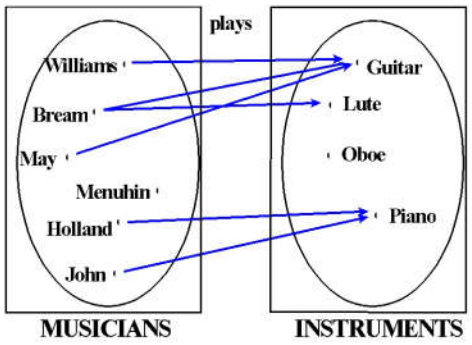
    Austria speaks German will be **true**

TO-SET AND FROM-SET

- For a relation X↔Y between the elements of the sets X and Y, then we say

  X is the *from-set* (or *source)*, and
  Y is the *to-set* (or *target*)

- Consider the relation *plays*, as illustrated alongside



- In the *plays* relation as shown,
  - the from-set has some element(s) not related to any element of the to-set ; likewise
  - the *oboe* of the to-set is not apparently played by any element of the from-set

DOMAIN AND RANGE

- When dealing with relations, we are interested, usually, only in those elements in the from-set and to-set which are actually related

- The subset of elements in the from-set which are related to *at least one* element of the to-set is called the **domain** of the relation

  e.g. the **domain** of the *plays* relation is:
  {Bream, Holland, John, May, Williams}

- In like manner, the subset of elements in the to-set, which are related to some element in the from-set, is called the **range** of the relation

  e.g. the **range** of the *plays* relation is: {Guitar, Lute, Piano}

- Usually we abbreviate *domain* and *range* to **dom** and **ran**, so the above examples can be written:

  dom *plays* = {Bream, Holland, John, May, Williams}
  ran *plays* = {Guitar, Lute, Piano}

RELATIONAL INVERSE

- It is often useful to look at a relation 'the other way round'

- The "infix" *_plays_* relation shows which instruments are played by particular musicians

- To show which musicians play particular instruments, we could define an 'other way round' infix relation called *_is-played-by_*

    - as part of the *_plays_* relation we have:
      Holland plays piano
      and so would expect:
      piano is-played-by Holland

- By reversing ***all*** the ordered-pairs in a relation we obtain another relation known as the ***inverse*** of the original relation

- Thus *_is-played-by_* is the inverse relation to *_plays_* and this is written symbolically as:

$$\_is\text{-}played\text{-}by\_ = \_plays\_^{-1} \qquad (\text{ or, } \_plays\_^{\sim})$$

RELATIONAL IMAGE

- If we are given any subset of the *domain* elements of a relation, then the subset of corresponding values from the *range* of the relation is given by the *relational image*

- From our *plays* relation, the set of instruments played by *Williams* or *Bream* is {*guitar*, *lute*} and this can be written symbolically, using the *relational image*, as

$$plays(\{Williams, Bream\}) = \{guitar, lute\}$$

- As a further example, for the *speaks* relation, an expression which specifies the languages spoken in France and Switzerland is:

$$speaks(\{France, Switzerland\})$$

RELATIONS ARE SETS

- Since a relation is a set of ordered-pairs it can be manipulated with the usual set operations:

- For the relation *plays* where:

$$plays : \text{MUSICIANS} \leftrightarrow \text{INSTRUMENTS}$$

if we add the extra information that *Oistrakh* plays the *violin*, we can write:

$$plays' = plays \cup \{(Oistrakh, violin)\}$$
$$\text{or,} \quad plays' = plays \cup \{Oistrakh \mapsto violin\}$$

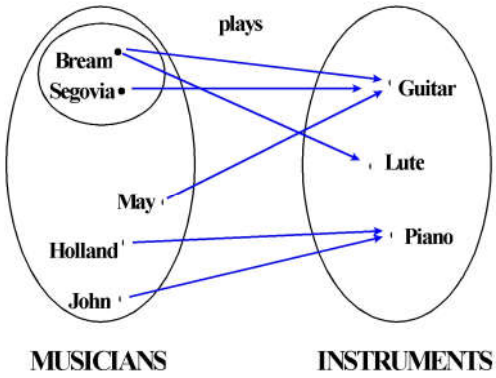
Here the prime annotation $'$ indicates an "updated" value

DOMAIN RESTRICTION

- It is often convenient to consider a relation restricted to only *part* of the original domain

- Consider the *plays* relation shown below where:

$$Classical\_musicians == \{Bream, Segovia\}$$



- If we are only interested in that part of the domain which includes the set of "classical" musicians, we could define a smaller (restricted) relation called, say, *Classical_plays* by:

$$Classical\_plays == Classical\_musicians \lhd plays$$

DOMAIN RESTRICTION

- Note that:

  $Classical\_musicians \lhd plays \quad \Leftrightarrow$

  $(Classical\_musicians \times INSTRUMENTS) \cap plays$

- $\lhd$ is the *domain restriction* operator and can be defined by:

  $$S \lhd R = = \{x{:}X; y{:}Y \mid x \in S \land x \mapsto y \in R\}$$

  where **dom** R : X and **ran** R : Y

- *Domain restriction* restricts the ordered-pairs of a relation so that only those pairs where the first element is contained in a particular, restricted, set of the original domain are included

DOMAIN CO-RESTRICTION

- Suppose further that the original domain of our relation, *plays*, comprises two **disjoint** sets (these could be *Non_Classical_musicians* and *Classical_musicians*) then if, as above, *domain restriction* has been used to isolate one of those two disjoint sets, the other disjoint set is given by *domain co-restriction* (aka *domain subtraction* or *domain anti-restriction*)

- If we assume that the musicians of our *plays* relation are either "classical" or "pop" musicians but **cannot be both** then domain co-restriction allows us to restrict the *plays* relation to *non-classical* musicians of the original domain thus:

$$Classical\_musicians \triangleleft plays$$

- Which is equivalent to something like:

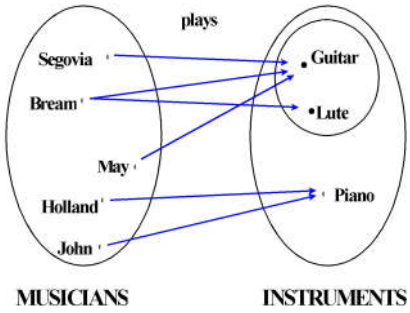
$$plays \setminus (Classical\_musicians \times INSTRUMENTS)$$

RANGE RESTRICTION

- To complement the *co-restriction* and *restriction* operations for *domains* there are similar operations that apply to *ranges*

- Suppose we wish to concentrate our attention on the set of 'plucked' instruments where

$$Plucked\_instruments = = \{Guitar, Lute\}$$



- To confine the *plays* relation to those ordered-pairs whose second members are in the *Plucked_instruments* set, we write:

$$plays \triangleright Plucked\_instruments$$

which is equivalent to:

$$(\text{MUSICIANS} \times Plucked\_instruments) \cap plays$$

RANGE CO-RESTRICTION

- *Range co-restriction* (aka *range subtraction* or *range anti-restriction*) draws attention to those pairs in a relation where the second elements are NOT members of some set of interest in the range of the relation

- Thus in our *plays* relation where we have separated the members of *Instruments* into *Plucked_instruments* and others:

  - the expression:

    $$plays \rhd Plucked\_instruments$$

    restricts the relation to those pairs where the second element is neither *Guitar* nor *Lute*

- $plays \rhd Plucked\_instruments \Leftrightarrow$

  $plays \setminus (MUSICIANS \times Plucked\_instruments)$

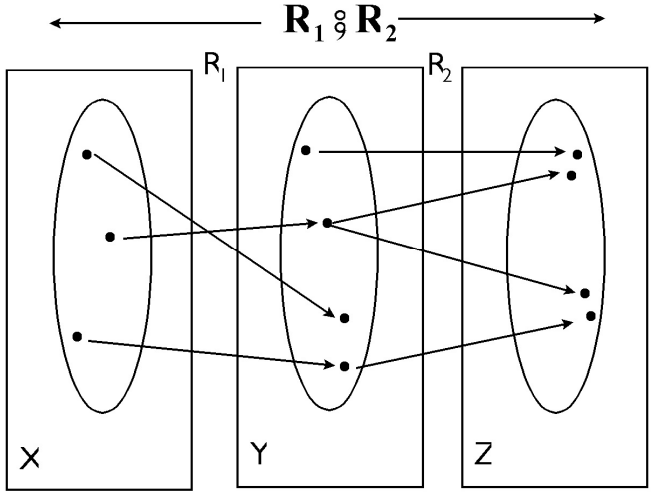
RELATION COMPOSITION

- If we have two relations R_1 and R_2 where the to-set of R_1 is of the same type as the from-set of R_2 then we can form a third relation R_3, say, which is the *composition* of R_1 and R_2:

$$R_1 : X \leftrightarrow Y \text{ and } R_2 : Y \leftrightarrow Z$$

- The relation R_3, formed by the composite relation "R_1 then R_2", is called the *forward composition* of R_1 with R_2 and is denoted by:

$$R_1 \fatsemi R_2 : X \leftrightarrow Z$$



- The symbol ⨟ is often called the *fat semi-colon*

REPEATED COMPOSITION

- If a relation relates domain values of some type to range values of the **same** type, the relation is said to be *homogeneous*

- The composition of homogeneous relations is often called *repeated composition*

- Clearly any homogeneous relation can be composed with itself (*self-composition*):

  if $R : X \leftrightarrow X$ then $R \fatsemi R : X \leftrightarrow X$

- If we imagine *_borders_* to be a relation between countries which share a border then:

  *_borders_* : COUNTRY $\leftrightarrow$ COUNTRY
  (e.g. France *borders* Germany)

- Writing the composition *borders* $\fatsemi$ *borders* implies two countries each share a border with a third:

  France *borders* $\fatsemi$ *borders* Austria

REPEATED COMPOSITION

- Such repeated composition is often abbreviated with a *power* symbol:

  - France *borders* ^{3} Hungary

    is equivalent to:

    France *borders* ⨾ *borders* ⨾ *borders* Hungary

    (which is, itself, a shortened form of:

    France *borders*
    Germany *borders*
    Austria *borders* Hungary)

- In general, *x* R $^{+}$ *y*, implies there is a repeated composition of relation _R_ which relates *x* to *y*

  For example: "France *borders*$^{+}$ India" implies France and India are on the same landmass

IDENTITY RELATION

- The *identity* relation is symbolized by **id** and maps all elements of a set onto themselves:

$$\mathbf{id}\ X = = \{ x : X \bullet x \mapsto x \}$$

e.g. if $X=\{p, q, r\}$
then $\mathbf{id}\ X = \{(p, p), (q, q), (r, r)\}$
or $\mathbf{id}\ X = \{p \mapsto p,\ q \mapsto q,\ r \mapsto r\}$

- It is, perhaps, worth noting that it is possible to define some of the operations on relations in terms of the identity relation:

e.g. $S \lhd R = = (\mathbf{id}\ S) \fatsemi R$

and, $R \rhd S = = R \fatsemi (\mathbf{id}\ S)$

SUMMARY of SYMBOLS

- P, Q, S are sets; p : P; q : Q and R : P $\leftrightarrow$ Q

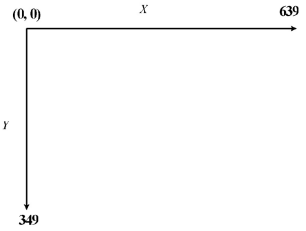
P × Q	*the set of ordered-pairs of elements from P and Q respectively*
P $\longleftrightarrow$ Q	*the set of relations from P to Q;* (*same as* $\mathbb{P}(P \times Q)$)
p R q	*p is related by R to q;* (*equivalent to* $(p, q) \in \_R\_$)
$p \mapsto q$	*ordered pairing of p and q*; (*equivalent to* (p, q))
$\{p_1 \mapsto q_1, p_2 \mapsto q_2,$ ..... $p_n \mapsto q_n\}$	*the relation* $\{(p_1, q_1), (p_2, q_2),...,(p_n, q_n)\}$ *relating* p_1 *to* q_1, p_2 *to* q_2, ..., p_n *to* q_n
dom R	*the domain of a relation*
ran R	*the range of a relation*
R^{-1}	*the inverse of a relation R;* (*also sometimes written* $R^{\sim}$)
R (⦇ S ⦈)	*the relational image of S in R*

- Below Q and S are sets; R, R_1, R_2 are relations

S ◁ R	*the relation R, domain restricted to the set S*
R ▷ S	*the relation R, range restricted to the set S*
S ⩤ R	*the relation R, domain co-restricted to the set S*
R ⩥ S	*the relation R, range co-restricted to the set S*
R_1 ⨾ R_2	*the (forward) composition of* R_1 *with* R_2
R^+	*the repeated self-composition of R with itself*
id Q	*the identity relation on Q*

EXERCISES

1. To show graphics on a PC we must be able to identify particular points on the VDU screen of the PC. Points (pixels) on the VDU screen are specified by their coordinates. The EGA graphics adaptor can handle a set of coordinates ranging from 0 to 639 horizontally and 0 to 349 vertically with the origin of the system in the top left-hand corner



If X and Y are sets defined as: $X == \{x:\mathbb{N} \mid x<640\}$;
$Y == \{y:\mathbb{N} \mid y<350\}$
write an expression in terms of X and Y which yields all possible pixels on the screen.

2. If $A = \{1, 2, 3, 4\}$, $B = \{2, 4, 6, 8, 10\}$ and $C = \{2, 3, 5, 7\}$ write down the following:

   (a) $\mathbb{P}C$
   (b) $A \times C$
   (c) $\#B$
   (d) $\#A$
   (e) $\#(A \times B)$

3. Given the sets A and B defined by: $A == \{p, q\}$; $B == \{1, 2, 3\}$
   write down the values of:

   (a) $\mathbb{P}A$
   (b) $\mathbb{P}B$
   (c) $\mathbb{P}(\mathbb{P}A)$
   (d) $\mathbb{P}A \cap \mathbb{P}B$
   (e) $A \times A$
   (f) $A \times B$
   (g) $A \times A \times A$
   (h) $\#(\mathbb{P}A \times B)$
   (i) $\#(A \times \mathbb{P}B)$
   (j) $\#(\mathbb{P}A \times \mathbb{P}B)$
   (k) $\{S:\mathbb{P}A \mid \#S=1\}$
   (l) $\{(x, y):B\times B \mid x+y\leq 2\}$

4. Given $R = \{(p, 2), (q, 3), (r, 4)\}$ and $S = \{(1, x), (2, y), (3, z)\}$ write down:

   (a) $\#R$
   (b) **dom** R
   (c) **ran** R

(d) **dom** S
(e) **ran** S
(f) R^{-1}

5. Use the definition of the inverse of a relation to argue the truth of the following statements:

(a) $(R^{-1})^{-1} = R$
(b) **dom** R = **ran** (R^{-1})
(c) **ran** R = **dom** (R^{-1})

6. Assume the following definitions for the relations *plays* and *worksby* over given sets PEOPLE, INSTRUMENT and ACTION (it should be obvious which elements go with which sets):

plays = = {(John, piano), (Williams, guitar), (Kennedy, violin), (Armstrong, trumpet), (Lennon, flute), (Lennon, piano), (Starr, piano)}
worksby = = {(piano, hammering), (guitar, plucking), (harpsichord, plucking), (trumpet, blowing), (flute, blowing), (violin, bowing), (violin, scraping)}

(a) What are the **domain** and **range** of *plays* and *worksby*?
(b) What are the **types** of *plays* and *worksby*?
(c) Write down $plays^{-1}$ and $worksby^{-1}$.
(d) What are the **domain** and **range** of $plays^{-1}$ and $worksby^{-1}$?
(e) Write out the contents of each of the following relations

(i) $\{piano, harpsichord\} \lhd worksby$?
(ii) $plays \rhd \{piano, violin\}$
(iii) $plays \rhd\!\!= \{piano\}$
(iv) $worksby \rhd\!\!= \{bowing, scraping\}$

7. Assume [COUNTRY] (the set of all countries) and [LANGUAGE] (the set of all languages) as base types with *speaks* as a relation between them (see chapter notes).

(a) Express the fact that *Latin* is not the language of any country.
(b) Express the fact that, in Switzerland, there are, officially, four languages spoken (they happen to be *French, German, Italian* and *Romansch* **but** assume that this fact is not known).

8. The set of *Users* of a computer system is the union of two other sets: *Normal_users* and *Privileged_users*. The association between the computer-users and the files that they own

is modelled by a relation *owns* over the two sets *Users* and *Files*. Write down an expression to represent the set of files owned by the members of the set *Privileged_users*.

9. Suppose *owns* and *can_read* are relations over USER × FILE. If current values are:

owns = {(Roberts, archive), (Wilson, tax), (Jones, old), (Roberts, summary)}
can_read = {(Roberts, archive), (Wilson, archive), (Wilson, tax), (Jones, tax), (Roberts, summary), (Jones, old), (Jones, archive)}

indicate which of the following predicates are *true* and which are *false*:

(a) *Roberts owns archive*
(b) ¬(*Roberts owns summary*)
(c) *(Jones, tax)* ∈ *owns* ∧ *(Roberts, archive)* ∈ *can_read*
(d) *owns* ⊂ *can_read*
(e) *#owns* = 7
(f) **dom** *owns* = **dom** *can_read*
(g) *Roberts owns archive* ∧ *(Roberts, archive)* ∈ *can_read*

10. If R_1, $R_2 : \mathbb{N} \leftrightarrow \mathbb{N}$ are the relations:
$R_1 = \{ 1 \mapsto 4, 2 \mapsto 3, 3 \mapsto 2, 3 \mapsto 3, 4 \mapsto 1 \}$; $R_2 = \{ 1 \mapsto 2, 2 \mapsto 3, 3 \mapsto 4\}$
write out:

(a) $R_1 \fatsemi R_2$
(b) $R_2 \fatsemi R_1$
(c) $R_1 \fatsemi R_1$
(d) $R_1 \fatsemi R_1 \fatsemi R_1$
(e) $R_1 (\!| \{2, 3\} |\!)$
(f) $R_2 (\!|\{0\}|\!)$
(g) $R_1^{-1} (\!|\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}|\!)$
(h) $(R_1 \fatsemi R_2) (\!|\{3\}|\!)$
(i) $((\{2\} \lhd R_1) \fatsemi (R_2^{-1})) (\!|\{2\}|\!)$
(j) $(R_1 (\!|\{2\}|\!)) \setminus (R_2 (\!|\{2\}|\!))$
(k) $(R_1 \setminus R_2) (\!|\{2\}|\!))$

11. The genealogy of historical personages can be investigated or, at least, recorded, with computer assistance, and the binary relation is an ideal model for a repository of such genealogical information. Suppose [PERSON] is a suitable base type, and suppose that V, A, C, L, Vy, E, Aa, D, W, G and N are distinct members of it.

Our knowledge about their relationships may be modelled by the *infix* relation:

*_is_a_parent_of_* : PERSON ↔ PERSON

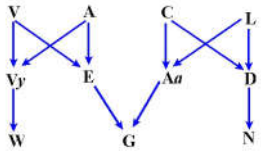
Note that the from-set and to-set are the same type.The genealogy of historical personages can be investigated or, at least, recorded, with computer assistance, and the binary relation is an ideal model for a repository of such genealogical information. Suppose [PERSON] is a suitable base type, and suppose that V, A, C, L, Vy, E, Aa, D, W, G and N are distinct members of it.

Our knowledge about their relationships may be modelled by the *infix* relation:

$$\_is\_a\_parent\_of\_ : \text{PERSON} \leftrightarrow \text{PERSON}$$

Note that the from-set and to-set are the same type.

The accompanying diagram represents a possible value of *_is_a_parent_of_*.



That V is a parent of E is illustrated by the arrow from V to E and
is expressed more formally by the expression: V *is_a_parent_of* E

(a) List the *domain* and *range* of the *is_a_parent_of* relation for the value shown
(b) List the relation *is_a_parent_of* when *domain restricted* to the set {V, E, G}
(c) List the relation *is_a_parent_of* when *range subtracted* by the set {V, E, G}

12. In the preceding chapter an example was given of a *speaks* relation.

If EU : ℙ COUNTRY and *speaks_in_EU* is a restricted form of *speaks* (relating to countries of the set EU only), write down an expression involving the *speaks* and *speaks_in_EU* relations.

13. Given that [PERSON] is a given set representing the set of all people, and we have the relations: *has_father*, *has_mother, has_parent* : PERSON ↔ PERSON
then the relation *has_sibling* (brother or sister) can be defined in terms of *has_parent* thus:

$$has\_sibling : \text{PERSON} \leftrightarrow \text{PERSON}$$
$$has\_sibling = (has\_parent \fatsemi has\_parent^{-1}) \setminus \mathbf{id}\ \text{PERSON}$$

(a) Use composition of relations to describe the *has_grandparent* relation in terms of the *has_parent* relation.
(b) If a person's first-cousin is defined to be a child of the person's aunt or uncle describe the *has_first_cousin* relation symbolically.
(c) Write down a definition of the relation *has_ancestor* in terms of *has_parent*.

14. Using the following relations *brother_of* and *mother_of*:

brother_of == {(John, Lynne), (Mike, Sue), (David, Mary)}
mother_of == {(Lynne, Chris), (Lynne, Matthew), (Sue, Paul)}

(a) verify that $(brother\_of \fatsemi mother\_of)^{-1} = (mother\_of^{-1} \fatsemi brother\_of^{-1})$

(b) What sort of blood relationship is defined by $(brother\_of \fatsemi mother\_of)^{-1}$?

15. The teaching rooms in a university are booked to different courses. Only Science courses can book rooms which are laboratories. Using given sets:

[ROOM]	the set of all possible rooms,
[SESSION]	the set of all possible sessions for which a room may be booked, and
[COURSE]	the set of all possible courses,

write formal expressions for the following:

(a) the relation *book* between rooms and courses
(b) the relation *book* restricted to laboratories
(c) the relation *book* restricted to Science courses
(d) a further relation *make* which links time-sessions to courses
(e) a composition of *make* with *book* (consider the need for inverses) to give the time slots when rooms are booked.

16. Suppose [BOOK] and [PERSON] are given sets for a Library system. Suppose, further, that *_lent_to_* is a relation such that *x lent_to y* means that the book *x* has been loaned to the person *y* (NB we shall see, later, that this relation is of a special sort).

(a) Write down an expression for the set of library books on loan to a person, *p*.
(b) If no borrower is allowed to have more than 8 books on loan, write down an expression restricting the number of loans for a borrower *p*.

17. Suppose R is a relation and suppose the domain of R is restricted to the elements of some set S using $S \lhd R$. Write down an expression connecting the *relational image* of R with the *restricted domain* of R.

18. A university operates a modular degree scheme such that students enrolled at the university are able to register for a selection of modules from a large menu. The choice is not entirely "free" and registrations for modules are subject to certain constraints. It is important that the administrators keep track of which students are doing which modules.

Basic types are: [PERSON, MODULE]

which, respectively, are the sets of people and modules that might ever be in the system.

If the ordered pair (p, m) where p is a student and m a module, represents the information that the student p is taking the module m, we might represent the entire modular degree scheme by the relation *taking*, where: $taking : \text{PERSON} \leftrightarrow \text{MODULE}$

The known constraints are:

- those who are registered for modules must be enrolled as students;
- the modules taken by students must be *bona fide* degree modules at the university;
- the greatest number of students allowed to take any module is *maxNum*

The following sets are to be used to model the system:

$students : \mathbb{P}\ \text{PERSON}$	the set of students currently enrolled at the university, and
$degModules : \mathbb{P}\ \text{MODULE}$	the set of modules currently in the modular degree scheme.

And *maxNum* will be modelled as a natural number (i.e. be of type $\mathbb{N}$).

(a) Write down a symbolic expression for the set which describes the rather unusual situation where every possible person is taking every possible module?

(b) If $firstYear : \mathbb{P}\ \text{PERSON}$ represents the set of all first-year students, create a symbolic expression to define the set of all first-year students who are registered for module m.

(c) Write an expression which gives all degree modules which have **no** students registered.

(d) Derive the set of students who are registered for module m

(i) by using *range restriction* on the relation *taking*;
(ii) by using the *relational image* concept.

(e) Write symbolic expressions which reveal the connections between the sets

(i) *students* and *taking*;
(ii) *taking* and *degModules*

(f) Write an expression for the set of all students who are registered for at least one module which student s is taking.

19. Certain medicines may be given to patients diagnosed as suffering from particular illnesses. Suppose we attempt to model this system using the following given sets:

[PATIENT]	:	the set of all people who might ever be patients;
[ILLNESS]	:	the set of all illnesses from which people may suffer;

[MEDICINE] : the set of all medicines that might be used to treat illnesses.

Since

- patients may suffer from more than a single illness;
- many patients can suffer from the same illness;
- a particular illness may be treated with different medicines; and,
- a particular medicine can be used to treat many illnesses,

it seems likely that the following sets of ordered pairs could prove useful as a basis for modelling:

$$complaints : \text{PATIENT} \leftrightarrow \text{ILLNESS}$$
$$treatments : \text{ILLNESS} \leftrightarrow \text{MEDICINE}$$

(a) By referring to the simple system modelled above

(i) state what sort of mathematical structure has been used to model the two sets *complaints* and *treatments*; and

(ii) draw a suitably labelled diagram which includes representations of the sets PATIENT, ILLNESS, MEDICINE, *complaints* and *treatments*.

(b) If *medication*s : PATIENT ↔ MEDICINE is a set of ordered pairs matching patients with the medicines that they are taking, write down a symbolic expression showing how *medication*s can be derived from the two sets *complaints* and *treatments*.

(c) If our model needs to incorporate extra information to show that particular medicines should **not** be prescribed to treat certain illnesses, we might declare:

$$not\_permitted : \text{ILLNESS} \leftrightarrow \text{MEDICINE}$$

where $(i, m) \in not\_permitted$ only when illness i should **not** be treated with medicine m.

(i) If *banned_medications* : PATIENT ↔ MEDICINE shows those medicines patients must **not** take, write a symbolic expression to derive *banned_medications* from sets defined above.

(ii) Write down a symbolic expression for the set containing pairs of medicines which may "clash", in the sense that the first medicine of the pair may be used to treat a particular illness while the second is forbidden for that illness.

(iii) Write down a symbolic expression for the set containing pairs of illnesses such that the first illness in the pair may be treated by some medicine while the second in the pair should not be treated by the same medicine.

(d) If at least one of the medicines m_1, m_2, m_3 is being taken by the patients in a particular set, write down a symbolic expression for that set using

(i) relational image
(ii) domain restriction
(iii) range restriction

(e) Create a symbolic expression which gives the set of medicines appropriate for illnesses i_1, i_2, i_3, or i_4 if the medicines are also suitable for patients p_1, p_2 and p_3.

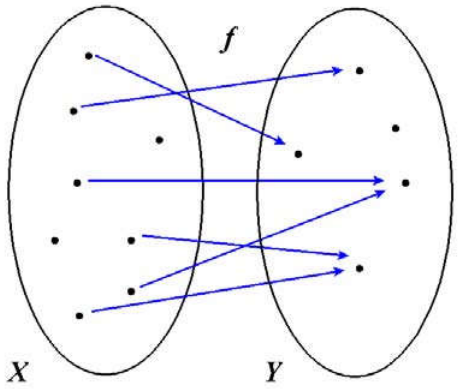
FUNCTIONS

This page is intentionally blank

☺

INTRODUCTION

- A *function* is a special type of *relation*, restricted so that any member of the *from-set* maps to, **at most**, one member of the *to-set*:



- The function *f* shown above from the set X to the set Y is declared by:

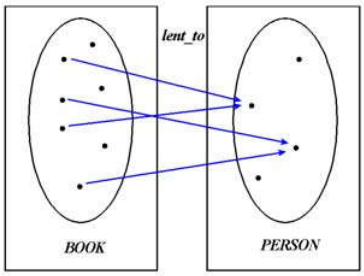
$$f : X \nrightarrow Y$$

and read as "the function *f* from X to Y"

- This is equivalent to a relation *f* from X to Y (f:X$\leftrightarrow$Y) but **restricted** so that, for each $x \in$ X, *f* relates that value *x* to, **at most**, one value *y* where $y \in$ Y

PARTIAL FUNCTIONS

- It is usual for there to be a rule that, for a Library, any book may only be *on-loan* to one person at a time



- The *lent_to* function illustrated would be declared by:

  $$lent\_to : BOOK \nrightarrow PERSON$$

  which denotes the set of all functions from the set BOOK to the set PERSON

- The **type** of *lent_to* is: $\mathbb{P}\,(BOOK \times PERSON)$

- Note that the domain of *lent_to* is only a subset of BOOK and this makes *lent_to* a *partial* function

- Partial functions are the most general sort of function

TOTAL FUNCTIONS

- If the domain of a function is the whole of the source (or from-set), the function is said to be *total*

- The symbol for a total function is similar to that for a partial function but without the transverse bar

- The declaration: $f : X \rightarrow Y$
  is read as "f is the total function from X to Y"

- Examples of total functions:

$$age : \text{PERSON} \rightarrow \mathbb{N}$$
$$has\_mother : \text{PERSON} \rightarrow \text{PERSON}$$

- Note: $X \rightarrow Y == \{ f : X \nrightarrow Y \mid \mathbf{dom}\, f = X \}$
  - by implication total functions are special cases of partial functions

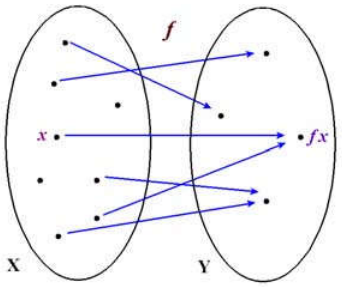
FUNCTION APPLICATION

- If ($f : X \pfun Y$) and ($x \in \mathbf{dom}\ f$) then we know there will be *at most* one value in **ran** f which will result from the mapping f being applied to the domain value x

- The *value* resulting from the application of a function f to x (its *argument*) is often written as:

  $\boldsymbol{f\,x}$ (or, sometimes, $f(x)$)

  $\boldsymbol{f\,x}$ is read as "f of x"
  or "f applied to x"



- For example, if $lent\_to$: BOOK $\pfun$ PERSON
  b : BOOK
  $b \in \mathbf{dom}\ lent\_to$

  then $lent\_to\ b$ represents the person to whom the book has been lent

- NB: $f\,x = f\,(\!|\,\{\,x\,\}\,|\!)$

FUNCTIONAL OVERRIDING

- Suppose a value of the *lent_to* function is:

$$\{029 \mapsto tom,\ 523 \mapsto sam,\ 109 \mapsto sam,\ 022 \mapsto jim\}$$

  where the first value in each pair is a unique book identifier and the second is the person to whom the book has been loaned

- Suppose, further, that *updates* is a set of pairs to be incorporated into *lent_to*:

$$updates = \{\ 029 \mapsto jim,\ 427 \mapsto tom\ \}$$

  which means:

  - book *029* has been transferred to *jim*, and
  - book *427*, previously on the shelves, has been loaned to *tom*

FUNCTIONAL OVERRIDING

- If *lent_to′* is the value of the function after amendment by *updates*, then *lent_to′* should contain all the pairs from *updates* and any of the original pairs from *lent_to* that do not conflict with those in *updates* - meaning those pairs from *old* which do not have the same first member as a pair from *updates*

- Hence, we recover:

  $lent\_to' = \{029 \mapsto jim, 427 \mapsto tom,$
  $523 \mapsto sam, 109 \mapsto sam, 022 \mapsto jim\}$

- Symbolically: $lent\_to' = lent\_to \oplus updates$

  where $\oplus$ is the sign to represent "overriding" or updating

- Since *functions* are *relations* (and, hence, *sets*) the usual set operations apply and *overriding* may be expressed, more clumsily, by:

  $lent\_to \oplus updates = ((\mathbf{dom}\ updates) \lhd lent\_to) \cup updates$

FUNCTION DEFINITION

- *Declaring* a function does **not** *define* its members

- Since a function is a set, we may use both set enumeration (provided there are not many maplet-pairs) and set comprehension to define a function explicitly

  e.g. $pos\_int\_sqrt == \{n : \mathbb{Z} \mid n \geq 0 \bullet n^2 \mapsto n\}$

- An alternative (and, often, shorter) form is provided by *lambda abstraction*

  e.g.: if $square == \{(0,0),(1,1),(2,4),(3,9),(4,16)\}$

  then, using *set comprehension*:

  $$square == \{n : \mathbb{N} \mid n \leq 4 \bullet n \mapsto n^2\}$$

  or, using *lambda abstraction*:

  $$square == \lambda\, n : \mathbb{N} \mid n \leq 4 \bullet n^2$$

FUNCTION DEFINITION

- By definition, a lambda expression describes a function, and, hence
  - the mapping is implicit
  - the set braces may be omitted

- Another example:

  if $total_{m<n}$ represents the function that accepts two non-negative integers and maps the pair to their sum, then we may define $total_{m<n}$ by:

  $$total_{m<n} == \lambda\, m, n : \mathbb{N} \mid m < n \bullet m + n$$

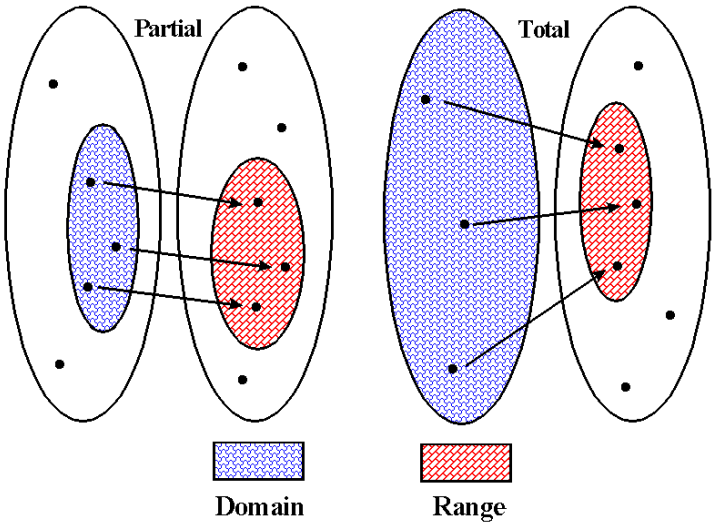
- The constraint bar is used only if needed

  e.g. $double == \lambda\, n : \mathbb{N} \bullet 2n$

  defines the infinite set which maps each non-negative integer to its double

OTHER CLASSES OF FUNCTIONS

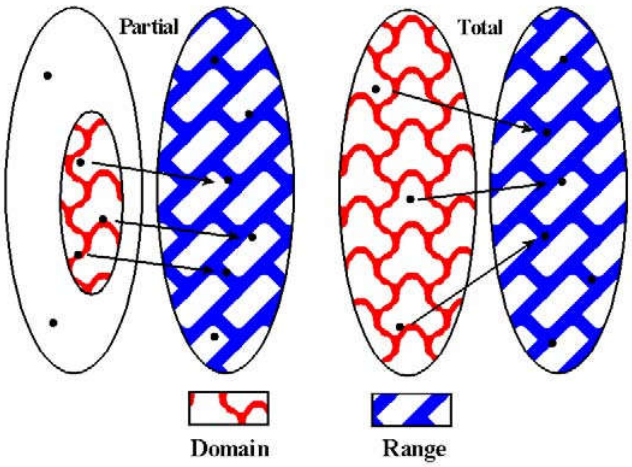
- As well as being partial or total, functions may be categorized as:
  - *injective* functions (or *injections*)



  - An *injection* is a 1:1 function: distinct elements of the domain map to distinct elements of the range (i.e. no two domain values map to the same range value)
  - A consequence is that an injective function will have an inverse which is also a function

OTHER CLASSES OF FUNCTIONS

- *surjective* functions (or *surjections*)

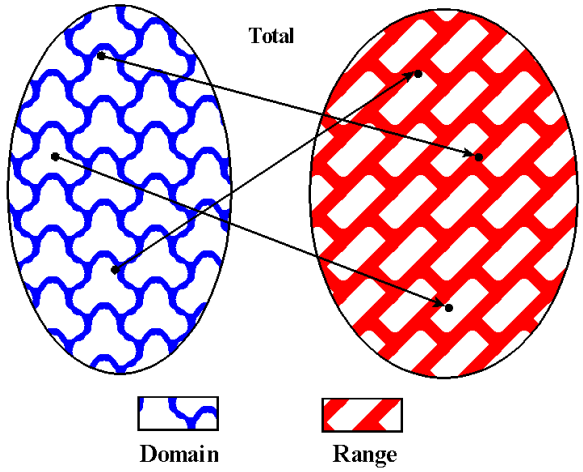


- Here, the range is the *whole* of the to-set (i.e. it 'fills' the to-set)

- *Surjective* functions are often referred to as *onto* functions

OTHER CLASSES OF FUNCTIONS

- *bijective* functions (or *bijections*)



- A *bijection* is a function that is a **total** 1:1 correspondence between the from-set and the to-set and, hence is both a *total injection* and a *total surjection*:
  - each member of the from-set maps to one and only one member of the to-set, and
  - every member of the to-set is associated with exactly one member of the from-set

CLOSING COMMENTS

- There are special arrow symbols associated with both partial and total versions of *injection*s, *surjections* and *bijections* but, in an effort to avoid confusion and 'symbol overload', we shall (at least for the present) not employ them

- Note that, in general, since functions are relations (and sets) the usual relation (and set) operations may be applied to functions BUT it should be observed that not all functions will have an inverse which is, itself, a function

- Whereas the set may be regarded as the fundamental simple atomic object of typed set theory, the relation is the fundamental complex object

- Simple sets, relations and functions form the cornerstones of formal specifications

SUMMARY OF SYMBOLS

$X \pfun Y$ *the set of partial functions from the set X to the set Y*

$X \fun Y$ *the set of total functions from set X to set Y*
$= = \{ f : X \pfun Y \mid \mathbf{dom}\, f = X \}$

$f\, x$ or $f(x)$ *the function f applied to x*

$f \oplus g$ *functional overriding*
$= = (\, \mathbf{dom}\, g \ndres f\,) \cup g$

$\lambda D \mid P \bullet E$ *function definition where P is a predicate constraining the values declared in D and E is an expression giving the function in terms of the values declared in D*

This page is intentionally blank

EXERCISES

1. Decide whether the following functions are *partial* or *total*:
   (a) A function called *halve* which maps integers to integers so that any value which is in **ran** *halv*e is exactly half of the corresponding value in **dom** *halve*.
   (b) A function called *passport* that maps people to passport numbers.
   (c) A function called *square* that maps integers to integers so that each value which is in **ran** *square* is the square of the corresponding value in **dom** *square*.

2. Decide what category of function best models each of the following:
   (a) The relationship between all the countries of the world and their capital cities.
   (b) The relationship between the countries of Europe and their capital cities.
   (c) The relationship between countries and their reigning monarchs.
   (d) The relationship between countries and their currencies.
   (e) The relationship between a month and its predecessor.
   (f) The relationship between a month and its successor.
   (g) The relationship between national flags and the countries to which they belong.

3. Decide whether any of the following relations may be functions:
   (a) *anagram* : letter_sequence $\leftrightarrow$ letter_sequence
   (b) *road_to* : town $\leftrightarrow$ town
   (c) *greater_than* : number $\leftrightarrow$ number
   (d) *has_number* : person $\leftrightarrow$ phone_number
   (e) *studies* : student $\leftrightarrow$ subject
   (f) *author_of* : person $\leftrightarrow$ book

4. Categorize each of the following relations as either a total function, a partial function or a relation which is not a function.

   (a) $much\_less\_than == \{ x, y : \mathbb{Z} \mid x < y - 99 \bullet x \mapsto y \}$
   (b) The size of the population of each country of the world as a relation from countries of the world to the set of integers.
   (c) The number of cars *owned* by a person as a relation from the set of people to $\mathbb{N}_1$.

5.
   (a) If function f is a *bijection*, describe the kind of function that is the inverse of f.
   (b) If function f is a *total injection*, describe the kind of function that is the inverse of f.

6. Suppose COMPANY is a given set: COMPANY = = {bt, ici, glaxo, shell}
and *share_price* is a function: *share_price* : COMPANY → ℕ
which models the share price of those companies which are members of COMPANY as a mapping from the set COMPANY to the set of natural numbers (in effect, the **range** of *share_price* gives the company share values in pence).

   (a) Suppose a regulatory committee bars *bt* from providing entertainment services over its phone-lines and the share price of *bt* consequently drops to 76 pence. If *sharePrices* is the value of the share_price function before the fall in *bt* shares and *newSharePrices* is the value after, write down an expression connecting *sharePrices* and *newSharePrices*.

   (b) If *double* is a function that maps any natural number onto a value which is twice the original value, write down a similar expression which will yield a doubling of the *bt* share price.

7. Explain using an example why $f \oplus g \neq (f \setminus g) \cup g$, for all f and g in $X \nrightarrow Y$

8. A vending-machine offers the following selections:

Drink	Price (pence)
Orange	25
Coffee	30
Cola	20
Tea	15

   (a) If f is the function mapping *Drink* to *Price*, categorize f.
   (b) Write a formal expression for the price of *Cola* being increased to 35p.
   (c) If the price of *Cola* is changed to 25p, how will this affect the functional model?

9. Suppose f and g are functions given by:

   $f = \{(a, x), (b, y), (c, z)\}$ and $g = \{(1, a), (2, a), (3, c)\}$

   (a) Determine $g \fatsemi f$ as a set of ordered pairs.
   (b) If $h = g \fatsemi f$, does h^{-1} exist?

10.

(a) A Library system is to be modelled using the given sets:

[BOOK]	which contains, as elements, all the possible copies of books which are likely to appear on the Library shelves;
[PERSON]	which contains, as elements, all people ever likely to be members of the Library;
[AUTHOR]	which contains, as elements, all people who are ever likely to be the authors of the books owned by the Library;
[TITLE]	which contains, as elements, all book titles likely to appear in the Library catalogue;

and the derived sets:

books	representing the set of book copies owned by a Library;
on_loan	representing the set of books currently on loan;
on_shelves	representing the set of books currently on the shelves;
borrowers	representing the set of people with books on loan from the Library;
members	representing the set of Library members.

Using these definitions, write expressions in set notation which are equivalent to the statements:

(i) *A book owned by the Library is either on the shelves or on loan*
(ii) *Only books owned by the Library can be on loan*
(iii) *Only Library members are allowed to borrow books*
(iv) *Borrowed books cannot still be on the shelves*

(b) Suppose *wrote* is a relation on the given sets AUTHOR and TITLE:

i.e. $\_wrote\_ : \text{AUTHOR} \leftrightarrow \text{TITLE}$

(i) What sort of values would be contained in the sets **dom** *wrote* and **ran** *wrote*?
(ii) *Thomas Hardy* wrote a book entitled *The Woodlanders*. Express that fact symbolically.
(iii) If *lent_to* is a (partial) function declared by: *lent_to* : BOOK $\rightarrow$ PERSON write down an expression for the set of library books on loan to a particular person *p*. If *p* is not a member of *borrowers*, what does this expression denote?

11. At a particular bank, a person is allowed to open no more than one account. Suppose *accounts* is a function relating each customer at the bank to their account, while *balance* is a function relating accounts and account-balances (which are always in whole numbers of pounds).

If we have given sets [PERSON] and [ACCOUNT] representing, respectively, all possible people who may ever open an account and all possible accounts that may ever be opened, then we may define:

$$accounts : \text{PERSON} \pfun \text{ACCOUNT}$$
$$balances : \text{ACCOUNT} \pfun \mathbb{Z}$$

(a) Explain why *accounts* and *balances* are **partial** functions.

(b)
- (i) Write a symbolic expression to state that all customer accounts will have balances.
- (ii) Create an expression which will yield a set containing ordered pairs where the first value in a pair is a *customer* and the second value is that customer's account balance.

(c) Write symbolic expressions to give:
- (i) all customers with accounts;
- (ii) the account which belongs to customer *c*.

(d) If the bank changes its rules so that any customer may have more than one account and customers can have joint accounts, modify the definition of *accounts* to reflect the change in the rules.

(e) For the revised model, write symbolic expressions to give:
- (i) the **number** of accounts owned by customer *c*;
- (ii) the **set** of customers who have at least one account *overdrawn*.

12. A university awards degrees with classifications defined by the free type DEGREE_CLASS where:

DEGREE_CLASS ::= *ordinary* | *pass* | *third* | *lower second* | *upper second* | *first*

If [STUDENT] defines the given set of all possible students, *comp_sci* denotes the set of final-year students graduating in *Computer Science* and the relationship showing the degree classification obtained by each student is modelled by:

$$final\_deg\_results : \text{STUDENT} \pfun \text{DEGREE\_CLASS}$$

(a) Write a symbolic expression which shows, for those graduating in *Computer Science*, who obtained which degree classification.

(b) Write a symbolic expression giving the **number** of non-*Computer Science* graduating students who were awarded *first* class degrees.

(c) Write a symbolic expression giving those students graduating in *Computer Science* who *failed* to get a *lower second* or better.

(d) Write a symbolic expression giving the complete set of final year *Computer Science* results after the External Examiner persuades the Examination Board to upgrade to a *third* class degree all those *Computer Science* students previously recommended for *pass* degrees.